ARPA

Command Systems Cybernetics

11

LEVEL III

78 07 14 001

(12)

Contract Period Covered by Report: (11) 1 March 1978 - 31 May 1978

(12) 57p.

(9) Quarterly rept. 1 Mar - 31 May 78,

Quarterly Research and Development Technical Report
(6) Spatial Data Management System,

Computer Corporation of America

Report Authors: (10) Christopher F./Herot
Jim/Schmolze

Research Division
Computer Corporation of America

617-491-3670

Sponsor: Defense Advanced Research Projects Agency Office of Cybernetics Technology

ARPA Order Number: 3487

ARPA Contract Number: (15) MDA 903-78-C-0122, ARPA Order-3487

Contract Period: 15 February 1978 - 30 November 1979

387 285

Gu

Table of Contents

1. INTRODUCTION

This report describes the second quarter of work on the design and implementation of a prototype Spatial Data Management ystem (SDMS). Spatial Data Management is a technique for organizing and retrieving information by enlisting the user's sense of spatiality through the use of high bandwidth, color, interactive, computer graphics.

These three months of the two-year effort were marked by the completion of the preliminary design phase and the beginning of detailed design. The results of the preliminary design are set forth in an associated report [HEROT, SCHMOLZE, CARLING, FARRELL]. That document presents the SDMS concepts in detail and describes the functional capabilities of the system under construction. Those capabilities are presented in Chapter 2 of this report in somewhat different form, organized around the capabilities provided to the users and administrators of the system. The reader unfamiliar with the SDMS concepts may wish to read Chapters 1 and 2 of the aforementioned Preliminary Design Document before proceeding with this report.

Progress toward the detailed design of the SDMS prototype is presented in Chapter 3 of this report which contains a description of the SDMS system architecture.

2. SYSTEM CAPABILITIES

The Spatial Data Management System provides each user with a graphical data space (GDS) consisting of nested surfaces of information, referred to as Information Spaces (I-Spaces). Each I-Space contains pictograms or icons which indicate to the user the location of particular items of information. An I-Space is stored in the computer as one or more image planes (i-planes) which are the actual bit arrays used to generate the displayed image. An I-Space may be composed of more than one image plane in order to allow it to be viewed at several levels of detail. An I-Space is thus a two-dimensional world over which the user can "fly", changing his altitude in order to control how much information he sees at one time.

In parallel to the graphical data space, the SDMS maintains a symbolic database management system (DBMS). Tuples in the DBMS may have corresponding icons in the GDS. Various capabilities are provided to allow the user to manipulate his view of the graphical data space and to allow him to create and maintain the GDS and its underlying database:

1. Retrieval Operations

   a. Scrolling and Zooming through the data space

   b. Rapid transit to a place in the data space

c. Symbolic retrievals (blinking and framing)

2. Modification Operations

a. Graphical annotation

b. Associating graphical and symbolic data

c. Creating ports and Information Spaces

d. Creating planes of image data (i-planes)

e. Defining Icon Class Description Language (IDCL)

f. Updating the symbolic database

Each of these operations is described in Section 2.3.

2.1 Database Administrator Functions

The capabilities listed above are arranged in the approximate order of difficulty of use. The system is designed to make graphical location and retrieval of information the easiest and fastest of all possible operations. However, it is intended that users be capable of performing at least a subset of the modification operations, so that they can organize their own personal data spaces. The operation of creating graphical representations of symbolic data has therefore been split into two parts. The association of a particular relation in the icon class description (which defines how each tuple in a relation will be displayed) will be a relatively simple task. The actual input of such icon class descriptions will be a more complex task, requiring greater knowledge of the workings of the system. It is anticipated that many SDMS installations will provide a library of such descriptions, maintained by the database administrator or some other designated individual.

The SDMS will provide for three classes of users: the Database Administrator (DBA), who has all capabilities and can delegate any of them to other users; users who are allowed to modify a particular graphical data space; and "casual" users who can only examine a GDS. Permission to invoke any of the system's capabilities can be specified for individual users in each class. Note that the issue here is not the privilege of writing a particular file, in the sense of Unix access control, but rather the complexity of the task (and the associated ability to get oneself into trouble) which can be performed.

This issue becomes especially important in the context of data spaces which are shared among a community of users -- such as a situation display. Certain users may have permission to modify the contents of the database, but the method by which the appearance of the information is defined (i.e. the Icon Class Description Language) must be carefully controlled in order that graphical representation which is employed remain consistent and intelligible to all of the users.

2.2 Input/Output Devices

A user interacts with SDMS through a variety of peripherals. The current implementation of the operators console contains:

1. 3 color CRT displays
2. 2 joy sticks
3. a data tablet
4. an alphanumeric keyboard

2.2.1 Displays

The three CRT displays are the means by which the user views the graphical data space. One of the displays always shows a view of the current I-Space at some chosen position. The other two displays serve either as navigational aids (maps of the data space) or to display menues of choices offered to the user.

The monitors are driven by a display processor capable of supporting three independent color video channels at a resolution of 480 x 640 pixels. One of the channels has sufficient memory to represent each pixel as an eight bit quantity, plus an overlay plane which can be used for cursors and text. The other two channels provide 4 bits per point, although they can be combined to yield a single 8 bit channel. The hardware is designed to permit future expansion to 16 bits per point. The values stored in each pixel's memory cell serve as indices into a 24 bit wide color lookup table, allowing any given image to contain 256 unique colors chosen from a universe of 16 million possible colors.

2.2.2 Joy Sticks

The joy sticks are the primary means through which the user commands the system to move about a graphical data space. They are of the spring return, displacement type. The user indicates his desire to travel in a particular direction by pressing in that direction on the appropriate joy stick. Both joy sticks can be operated in the x-y plane. One of the joy sticks also offers a third degree of freedom by twisting the handle. As motion through a graphical data space requires the specification of only three parameters (x-speed, y-speed, and zoom-speed), there is room for experimentation to determine the optimum input strategy from the human factors point of view.

2.2.3 Data Tablet

The data tablet serves many input functions:

1. Specifying a location to move to (rapid transit)

2. Command input (menu selection)

3. Annotation (drawing)

Many of these operations use the tablet in conjunction with one of the display screens -- the user moves the stylus on the tablet and watches the cursor move on the corresponding display.

In order to allow the one tablet to be used with the three displays, the system software splits the tablet into four "virtual tablets." One of these virtual tablets corresponds to a menu area drawn on the tablet's surface. The other three correspond to the large area in the center of the tablet and are selected by a set of three buttons mounted on the tablet. The user selects which screen he wishes to point at by depressing the corresponding button. This selection can also be accomplished under program control if the determination can be made by context, such as when the user is asked to give an argument to a command he has just issued.

2.2.4 Keyboard

While most operations will be performed with the joy sticks and the tablet, there will be times when the entry of alphanumeric information is called for. This is most important when interacting with INGRES, the symbolic database management system. It is envisioned that casual users of SDMS who are not users of INGRES will be able to carry on all of their interactions without the use of the keyboard.

2.3 Functions Provided to the User

This section describes each of the functions listed at the beginning of Section 2. It tells how the user invokes each one and which I/O devices are involved.

2.3.1 Scrolling and Zooming Through the Data Space

Moving through the graphical data space is the fundamental mode of operation in SDMS. It is available at all times, even while the user is in the process of executing another operation.

Scrolling and zooming are controlled by means of the joy sticks. One controls motion in the plane of the data space. Pushing the stick in any direction causes the user to move in that direction, with the data thereby scrolling across the screen in the opposite direction. The other joy stick controls motion perpendicular to the plane of data. Pushing the stick towards the screen moves the user closer to the data. The data in the Information Space gets larger and, if multiple image planes have been defined, more detail appears. Pulling the stick back has the opposite effect.

Special areas called ports can be defined in an I-Space. Zooming in on them causes a new I-Space to be entered. In this manner, I-Spaces can be nested to allow information to be partitioned in any convenient manner. The mapping of ports to I-Spaces is not limited to a hierarchy. A port can be defined to lead to any I-Space, and an I-Space may have more than one port leading to it.

2.3.2 Rapid Transit

While the user usually moves through the graphical data space by use of the joy sticks, there may be times when he wishes to get to some particular location without traversing all of the intervening territory. For this purpose, a rapid transit facility is provided. The system maintains a "world view map" of the top-level I-Space which is usually displayed as a navigational aid on one of the monitors. This map displays the user's current location on a map of the entire I-Space. By selecting the virtual tablet corresponding to that display, and pointing to a location on the map, the user can be quickly transported to that location. This is possible even if he is in some deeply nested I-Space, allowing a quick return to the top level.

If a world view map has been defined for some inferior I-Space and is currently displayed on one of the monitors, this same technique can be used to move to a point in that I-Space.

2.3.3 Symbolic Retrievals

For those occasions when the spatial location of a particular piece of information is not known, or a large amount of data must be searched, a symbolic query facility is provided. The query language of SDMS, referred to as SQUEL, is an extension of QUEL, the query language of INGRES. The extensions provide a facility for specifying graphical output of symbolic queries. The user can perform a retrieval operation in the usual fashion, but instead of storing the results in a new relation or printing them out, he can request that the associated icons be blinked or framed in a certain color. Where just one icon is involved, the user can ask the system to position him at its location. More sophisticated searches are possible by use of the association facility described in Section 2.3.5.

2.3.4 Graphical Annotation

Annotation is perhaps the simplest update operation which the user can perform. At any time the user can draw on the image plane merely by picking up the pen and drawing on the tablet. Unless he has specified otherwise, lines of the default color and width will be drawn, and at a sufficient rate to produce smooth curves. This facility can be used by the user to make notes about the data, call attention to certain items, or add new information.

At a slightly higher level of difficulty, a more sophisticated level of drawing is possible, and is described in Section 2.3.7.

2.3.5 Associating Graphical and Symbolic Data

The SDMS actually maintains two parallel databases, a graphical data space (GDS) and a symbolic database management system (DBMS). The DBMS in the prototype system is INGRES, a relational database system developed at the University of California at Berkeley [HELD, STONEBRAKER, WONG]. While some types of information are more suited to one type of database than the other, significant advantages accrue to having all of the data stored in both. For example, a database about ships may have a relation in the DBMS containing a tuple for each ship and an icon in the GDS for each tuple. This would give the user the option of browsing through the graphical data space or performing a query on the symbolic database.

In order to define the correspondence between the graphical and the symbolic data, an association mechanism is provided. There are two kinds of associations that may be defined: specific association and class associations.

Specific associations are intended for the case where both the graphical and the symbolic data already exist. For example, given a personnel database with a tuple for each employee, and an Information Space with a photograph of each one, a user would explicitly specify which photograph corresponded to each employee.

Class associations are used for creating an Information Space given a symbolic database. They depend on descriptions written in an Icon Class Description Language (ICDL) which define the graphical appearance of particular pieces of symbolic data. For example, in a database of ships, an icon could be drawn for each ship, with the icon's size being a function of its gross tonnage and its color being a function of its nationality. Given an ICDL description and a relation in the

DBMS, the user can put many icons into an I-Space and define their correspondences to tuples in the DBMS with a single command.

Writing the icon class descriptions themselves is a more difficult process, and is described in Section 2.3.8.

2.3.6 Creating Ports and Information Spaces

Ports are the mechanism through which the user can escape from the current I-Space into either another I-Space or some arbitrary Unix process. They are defined while the I-Space is on the display, as in normal operation. The user points at the "CREATE PORT" menu button, and then defines the location of two opposite corners of the port by pointing to them with the tablet. He then specifies whether traveling through the port should lead to a new I-Space, an old I-Space, or some Unix process. Entering the choice causes the connection to be established. If the port is to lead to a new I-Space, it is created and may be initialized by entering it by the usual process of zooming.

2.3.7 Creating Image Planes

An image plane contains the actual bit-map representation of an I-Space at some level of detail. When an I-Space is first created, it consists of only one image plane. If a user wishes to add a more detailed image plane, he zooms to the scale at which he wishes the transition between the two planes to occur and selects the "MAKE I-PLANE" menu button. This puts him into the new image plane. If desired, the old image plane can be copied into the new one before more detail is added.

Image planes can be automatically filled by making class associations or they can be manually filled by invoking the image plane editor, a more sophisticated version of the annotation program. Selecting the "EDIT" button on the tablet menu causes the world view map on the auxiliary monitor to be replaced by a menu of image plane drawing functions. These include:

1. Color selection

2. Line-type selection

3. Insertion of circles, rectangles and simple geometric figures

4. Flooding enclosed areas with a color

5. Input of text

6. Digitizing photographs

2.3.8 Defining Icon Class Descriptions

Icon class descriptions are composed of two parts. One part is a bit-map representation, drawn in a special I-Space with the programs mentioned above. The pictures are stored in such a way that the colors, scale, and location can be altered later when a copy of the bit-map is displayed. The second part of the description is a program written in Icon Class Description Language (ICDL) which defines how those parameters of color, etc. are determined from the data in a particular tuple. When the user makes a class association, the system draws an icon for each selected tuple, using the ICDL statements to determine how and where the icon is to be drawn.

2.3.9 Updating the Symbolic Database

The DBMS may be accessed through all of the standard QUEL commands. Changes to the DBMS which effect icons in the GDS are noted by the system and result in corresponding modification of the affected icons.

3. SYSTEM ARCHITECTURE

The set of facilities offered by SDMS has been explained in Section 2. This section will present a view of the implementation of SDMS by describing the component modules, their functions and the communication between them.

3.1 Modes of Operation

SDMS can operate in several modes. At all times, the main graphic screen will show a view of the GDS with motion about it being controlled by the joysticks. The normal mode of SDMS allows SQUEL commands to be entered at the keyboard. While in this mode, called SQUEL mode, the GDS editor will usually allow only annotation to i-planes. By depressing the "EDIT" menu button, the full capabilities of the GDS editor become available.

The second important mode of SDMS is intended for entering and editing Icon Class Description Language. SQUEL is not available in this mode, called ICDL mode. Normally, the GDS editor is available with full capabilities.

When the SDMS system is activated, a set of Unix processes is started. At any given point in time, most of the modules in these processes are dormant, awaiting input from one or more i/o devices or inter-process channels (in Unix, these channels are called "pipes"). For example, the annotation program is waiting for input from the virtual tablet for the main graphic screen so it can annotate the GDS. There are two possible causes for a module to be waiting for input. One cause is that the user simply has not entered any input. The other cause is that some other module has control of the input device, so this module does not receive any input from it. The modes of operation will be controlled by the allocation of the i/o resources, and are orchestrated by the Process Overseer.

The following example shows how the Process Overseer switches from SQUEL mode to ICDL mode. For the purposes of this discussion, the keyboard and the text display are referred to as a terminal. In actuality, they may be separate devices, with text being displayed through the graphics system. For SQUEL mode, the terminal is connected to the SQUEL processor, so SQUEL commands can be typed in and executed. The ICDL processor is operating simultaneously but since it has no terminal connected to it, it receives no commands and performs no actions. When the Process Overseer switches to ICDL mode, the terminal is disconnected from SQUEL and is connected to ICDL. Simultaneously, ICDL is also allowed to use an auxilary screen for posting menus. Now ICDL receives and executes commands while SQUEL lies operational but dormant. The allocation of graphic screens works in a similar fashion. The devices which control the mode of operation are a keyboard, text display, three graphic screens and four virtual tablets.

3.2 Controlling I/O Devices

A heavy emphasis is put on the role of i/o devices since they are the primary tool for determining the system's mode of operation. The control of the i/o devices is not complicated but it is important to the understanding of SDMS. The devices are:

a. 3 graphic screens

b. joysticks

c. 4 virtual tablets

d. terminal - keyboard and text display

The tablet will have a strip on the top devoted to a static menu. These commands are for changing mode, getting help, etc., and may be entered at any time. The rest of the tablet is devoted to the screen that it is currently pointed at. The SDMS views the tablet as four virtual tablets. One is for the static menu. The other three are for each graphic screen. The virtual tablet facility is implemented as part of the Unix i/o driver for the tablet.

When SDMS is started, all devices are allocated to the Process Overseer. The Process Overseer allocates the devices to the various modules as necessary for the required mode of operation. The allocation of a device is done by a function call from the Process Overseer to a module. The arguments of the call are the device identifier and a privilege value. This privilege will be a special value, designated by the Process Overseer, that is required when a module attempts to seize a device. The privilege value allows the right to seize a device to be passed from one module to another. This is useful, for example, when the Process Overseer gives access privilege for an auxiliary screen to the Navigator and the Navigator, in turn, passes that privilege to the module for Navigational Aids. The auxiliary screen can then be used for displaying a navigational aid.

When it is necessary to remove access to a device, the Process Overseer informs the module of the lost access. Unix is instructed to disconnect the device from whomever is currently connected to it. The access privilege is reset to a new value by the Process Overseer and given to the next module needing the device.

3.3 The Modules of SDMS

This section will summarize the functions of each of the modules of SDMS and will give a brief description of the algorithms employed. The communication between modules is explained in detail in Section 3.4.

3.3.1 Process Overseer

The Process Overseer will be the controlling module of SDMS. When SDMS is first activated, it initiates all the processes needed for full operation. It allocates the i/o devices and controls the mode of operation for the system.

The menu monitor will always respond to commands entered on the static menu by signaling the Process Overseer. Usually, these requests will change mode of operation. The Process Overseer performs this by re-allocating the i/o devices to the appropriate modules. A device is re-allocated using the following steps:

1. Inform the module who currently controls the device that it is losing the privilege of control.

2. Instruct Unix to disconnect the module from the device.

If the module has already done so, this has no effect.

3. Assign a new privilege value for the device.

4. Inform the module who is to receive control that it now has control and give it the privilege value.

The module can seize the device and begin processing whenever it is ready. If the device is a display, the module will typically fill it with the appropriate picture.

3.3.2 SQUEL Processor

The SQUEL processor receives typed input from the keyboard when SDMS is in its normal mode. SQUEL parses this input, recognizes the command and passes the parsed command to another module. The commands are either QUEL queries or association statements. The SQUEL processor also does a share of the work of maintaining the integrity of the GDS and it performs the interactions between INGRES and the graphic displays.

The association statements are simply parsed and then passed to the Icon Creation module. QUEL queries are a bit more complicated. SQUEL must recognize whether any pseudo-relations, such as BLINK or FRAME, are used, and if so, it must supply the values of these pseudo-relations to QUEL. SQUEL must also note any tuples that are either added, deleted or updated. These tuples are passed to the Icon Creation module which will determine what changes, if any, to the GDS are necessary for maintaining the integrity of the GDS.

3.3.3 Icon Creation

The module for Icon Creation has several responsibilities. It processes the association statement for both specific and class associations. For the processing of class associations, it will create icons from icon class descriptions and will insert them into the GDS. Along with the SQUEL Processor, it shares the responsibility of maintaining the integrity of the GDS.

The SQUEL Processor receives all association statements that are entered, parses them and upon recognizing them, passes them to the module for Icon Creation. This module records the association and, if it is a class association, creates icons for the specified tuples.

When a tuple of INGRES is either added, deleted or updated, the SQUEL Processor passes the tuple to this module. If that tuple was used to create an icon, or it should now be used as such, the icon is re-created. In this way, the icons that represent INGRES data are always truthfully represented.

3.3.4 ICDL Creating / Editing

Entering and editing icon class description language requires a separate mode of operation. This module interacts with a user to obtain either an icon class description or picture function.

An important feature of entering ICDL is that the user will be able to test out a description. By communicating with the SQUEL module, it will be able to take an icon class description, generate an icon from it and allow the user to view it in the GDS. Errors in the description can then be corrected and the test repeated.

3.3.5 GDS Editor

The GDS Editor is the primary tool for creating and editing the GDS. This module will get commands from the user via the menu monitor and will show its effects on the main graphic screen which views the GDS.

The changes to the GDS fall into two categories. The first affects the structure of the GDS by adding i-planes, ports, etc. These changes are passed to the Navigator who then updates the GDS. The second type of change is the actual coloring of the pixels on the i-planes. The coloring can be done by using the tablet and pallette, by filling in regions with a color, etc.. When these pixels are colored, this module passes the pixel coordinates and their new color to the Stager. The Stager will then update the i-plane.

There are two modes of operation for the GDS Editor. The full edit mode is the one just described. The second mode allows only annotations to the GDS with no coloring or structural changes. The mode is decided by the Process Overseer from the user's commands.

3.3.6 Icon Manager

The Icon Manager will manage the information relating icons and their location in the GDS. In fact, all objects which require space in the GDS will be stored in the Icon Manager's database. This includes icons plus text regions, update regions, etc. It will also perform the functions of blinking icons, framing icons and displaying text near icons.

The Icon Manager will be able to translate from icon to GDS coordinates and vice versa. It will also perform the "find space" service for adding icons to the GDS. This service takes a request for space at a location in the GDS and determines whether or not the space is available. If the space is not available, it will try to find space nearby.

Many temporary modifications to the GDS will be handled by the Icon Manager in conjunction with the Stager. These modifications include blinking an icon, framing an icon, temporarily displaying text in a text region, etc.. The Icon Manager will be responsible for initiating these modifications and for terminating them.

3.3.7 Navigator

The primary concern of the Navigator is to monitor the joysticks and respond to them by displaying a view of the GDS on the main graphic screen. It instructs the Stager what should appear on the display. It also instructs the Navigational Aid module as to which navigational aids to display on the auxilary screens and where the user is in the GDS so the navigational aids are updated correctly. The Navigator keeps track of precisely what portion of the GDS is being viewed by the user and it makes this information available to other modules needing it. Finally, changes to the structure of the GDS, such as adding new i-planes, are sent from the GDS Editor to the Navigator. The Navigator will perform the actual update the GDS.

3.3.8 Stager

The Stager receives commands from the Navigator instructing it as to which portion of the GDS is to be displayed on the main graphic screen. It performs the task of displaying the GDS. It also receives changes to i-planes from the GDS Editor and updates the i-planes accordingly.

Due to the large amount of information needed for the graphic screen, a large part of the Stager's efforts will involve managing the storage of the pixels in memory. Each i-plane is divided into tiles, where a tile would cover a small portion of a screen. The tiles are pulled from the disk into memory whenever the user "flies" close to them. When he is far from a tile, the tile is discarded so other tiles can be loaded into memory. In this way, the tiles on the screen plus a margin surrounding what is on the screen are always in the memory. Then motion can be shown smoothly about the i-plane.

3.3.9 Navigational Aids

The Navigator instructs the Navigational Aid module which aid to display and which screen to use. This module performs the actual displaying of the aid. The Navigator passes the current position of the user in the GDS to this module so the navigational aid is updated. The Navigational Aids module will also assist the Navigator in "rapid transit". When the user points to a position on a navigational aid screen, the Navigator goes there immediately. To perform this, the Navigator gets the screen coordinates of the position from Unix. The Navigational Aid module translates the screen coordinates to GDS coordinates and the Navigator goes there.

3.4 Function Calls Between Modules

Glossary

device-id	the id for an i/o device. These id's are provided by the system.
edit-mode-type	specifies mode of operation for the GDS Editor. Either full edit or annotate only.
GDS-coord	coordinate position in the GDS: <I-Space-id,x,y,i-plane#>
icdl-id	identifies an ICDL description - either an icon class or picture description.
icon-id	identifies a particular icon in the GDS.
icon-type	the types include picture, text region, update region, etc.
image-file	Unix file which stores an image plane.
i-plane-id	identifies an i-plane; consists of the I-Space-id and its index # in the I-Space.
I-Space-id	identifies an I-Space.
menu-description	specifies the items on the menu and how to draw it.
menu-id	identifies a menu.
modification-type	stager may modify tiles as they are staged in. The types of modification defined to date are for blinking, framing and display-

	ing text.
nav-aid-id	identifies a particular navigational aid.
nav-aid-type	type of navigational aid.
picture	includes a bit map plus the size of the picture (in pixels).
privilege	the privilege to seize a screen or tablet is an object that may be passed. This will probably be implemented with passwords.
screen-coord	position on a graphics sreen.
screen-pos	(x,y) position on a graphic screen.
screen#	identifies a particular graphics screen.
tile-id	identifies a tile from an image plane.
tuple-id	identifies a particular tuple in INGRES.
Unix-process	specifies how to call a particular Unix program.
x-ext	extent of a region in the x direction.
y-ext	extent of a region in the y direction.
z-ext	extent of a region in the z direction (# of i-planes).

<u>Process</u> <u>Overseer</u> to

1. GDS Editor

   a. set-edit-mode(edit-mode-type) - sets the mode of operation for the GDS Editor. The modes are either full-edit or annotate-only.

2. Navigator

    a. goto(GDS-coord) - starts the flying capability in the given position in the GDS.

    b. grant-privilege(device-id,privilege) - passes the privilege to seize a device.

    c. deny-privilege(device-id) - notification that the privilege to seize a device is being removed.

3. Menu Monitor

    a. grant-privilege(device-id,privilege) - passes the privilege to seize a device.

    b. deny-privilege(device-id) - notification the that privilege to seize a device is being removed.

4. Unix

    a. set-privilege(device-id,privilege) - sets the privilege for a device.

    b. release-device(device-id) - releases a device.

    c. connect-device(device-id,channel) - connects a device to an i/o channel.

d. disconnect-device(device-id,channel) - disconnects a device from an i/o channel.

<u>SQUEL</u> to

1. INGRES

   a. QUEL

2. Icon Manager

   a. get-icons-within-region(GDS-coord,x-ext,y-ext) - gets list of icons within given region of the GDS.

   b. display-text(icon-id,text) - displays the given text in the free text field for the given icon.

   c. erase-text(icon-id) - erases display text for the given icon.

   d. erase-all-text() - erases all currently displayed text

   e. blink(icon-id)

   f. unblink(icon-id)

   g. unblink-all()

h. frame(icon-id)

i. unframe(icon-id)

j. unframe-all()

3. Navigator

a. goto(GDS-coord) - used for rapid transit.

b. get-current-screen-pos() - returns current screen position in GDS plus size of screen.

c. screen-to-GDS-coord(x,y) - given a screen position, it returns the GDS-coord.

4. Icon Creation

a. link(tuple-id,icon-id) - link command

b. associate(I-Space-id,relation,icon-class,qualification,region) - associate command

c. update-tuple(tuple-id) - when tuples are updated, they are sent to Icon Creation to see if the GDS must be updated also.

5. Menu Monitor

a. display-menu(menu-description) - displays a menu and returns a menu-id. If the menu can be displayed, this routine succeeds, otherwise it fails.

b. read-menu-selection(menu-id) - returns menu selection and (x,y) position of point touched in the menu box. If the Menu Monitor had to delete the menu for some reason (if it lost access to the menu screen), this routine will fail.

c. read-point(device-id,privilege) - user points to a position on the screen, screen coordinates are returned. The privilege must be valid.

d. read-region(device-id,privilege) - user is prompted to point to a region (a rectangle). The screen coordinates of the diagonal corners of the region are returned. The privilege must be valid.

6. Unix

a. seize-device(device-id,prililege) - used to seize control of a device, returns id for usage with system calls. Privilege must be valid for success.

b. release-device(device-id,privilege) - releases a device. Fails if the privilege is incorrect.

c. point-tablet(device-id) - points the tablet to a particular screen. No privilege required.

<u>Icon Creation</u> to

1. Icon Manager

   a. reserve-space(GDS-coord,x-ext,y-ext,z-ext,icon-type) determines if the requested space is available, and if so reserves it. Returns icon-id. This routine can fail.

   b. find-and-reserve-space(GDS-coord,x-ext,y-ext,z-ext,icon-type) - finds space in the I-Space for the given planes for a rectangular volume of the correct size. It finds a space as close to the requested (x,y) as possible. It reserves the space and returns the icon-id. This routine can fail.

   c. add-picture(icon-id,picture) - adds a picture to the space reserved for the icon. If this space covers more than one plane, the picture is put onto all of the planes. This routine marks the icon as picture type. The picture should match the icon size exactly - no fitting performed.

d. add-text(icon-id,text) - puts text in space for icon. Similar to add-picture.

e. move-icon(old-icon-id,new-icon-id) - icon is moved from the old to the new. It retains the id from the old id & the new id is deleted

2. INGRES

a. QUEL

3. Icon Class Descriptions Database

a. get-icdl(icdl-id)

b. get-dependency-checks(icdl-id)

<u>ICDL Creation/Editing</u> to

1. Icon Class Descriptions Database

a. get-icdl(icdl-id)

b. add-icdl(icdl-id)

c. replace-icdl(icdl-id)

2. SQUEL

a. test-icdl(icdl-id,qualification,I-Space-id,relation) - this call allows the user to test an icon class. It instructs SQUEL to create a class association for only the given tuple, which will cause the creation of an icon from the ICDL. SQUEL then remembers where it is in the GDS, goes to the new icon and returns the id for the new icon.

b. erase-test(icon-id) - undos the effects of the test and returns to the original position in the GDS.

3. Menu Monitor

a. display-menu(menu-description) - displays a menu and returns a menu-id. If the menu can be displayed, this routine succeeds, otherwise it fails.

b. read-menu-selection(menu-id) - returns menu selection & (x,y) position of point touched in the menu box. If the Menu Monitor had to delete the menu for some reason (if it lost access to the menu screen), this routine will fail.

c. read-point(device-id,privilege) - user points to a position on the screen, screen coordinates are returned. The privilege must be valid.

d. read-region(device-id,privilege) - user is prompted to point to a region (a rectangle). The screen coordinates of the diagonal corners of the region are returned. The privilege must be valid.

4. Unix

a. seize-device(device-id,prililege) - used to seize control of a device, returns id for usage with system calls. Privilege must be valid for success.

b. release-device(device-id,privilege) - releases a device. Fails if the privilege is incorrect.

c. point-tablet(device-id) - points the tablet to a particular screen. No privilege required.

<u>GDS</u> <u>Editor</u>

1. Stager

a. read-color(x,y) - returns color of given point in current image plane. Note: x,y are in screen coordinates

b. write-color(x,y,color) - sets color of point at (x,y) to color. Note: x,y are in screen coordinates

2. Navigator

   a. make-i-plane(I-Space-id)

   b. make-port(GDS-coord,I-Space-id) - if a port can be made, it is done otherwise routine fails (ports only exist on bottom i-plane) The port connects to the givin I-Space.

   c. make-virtual-terminal(GDS-coord,unix-process) - used for connecting a port to a Unix process.

   d. delete-I-Space(I-Space-id)

   e. delete-i-plane(i-plane-id)

3. Icon Manager

   a. make-icon(GDS-coord,x-ext,y-ext,z-ext) - will fail if it overlaps another icon, otherwise returns icon-id.

   b. get-icon-id(GDS-coord)

   c. delete-icon(icon-id)

d. move-icon(old-icon-id,new-icon-id) - icon is moved from the old to the new. It retains the id from the old id & the new id is deleted

4. Menu Monitor

a. display-menu(menu-description) - displays a menu and returns a menu-id. If the menu can be displayed, this routine succeeds, otherwise it fails.

b. read-menu-selection(menu-id) - returns menu selection and (x,y) position of point touched in the menu box. If the Menu Monitor had to delete the menu for some reason (if it lost access to the menu screen), this routine will fail.

c. read-point(device-id,privilege) - user points to a position on the screen, screen coordinates are returned. The privilege must be valid.

d. read-region(device-id,privilege) - user is prompted to point to a region (a rectangle). The screen coordinates of the diagonal corners of the region are returned. The privilege must be valid.

5. Unix

a. seize-device(device-id,privilege) - used to seize control of a device, returns id for usage with system calls. Privilege

b. release-device(device-id,privilege) - releases a device. Fails if the privilege is incorrect.

c. point-tablet(device-id) - points the tablet to a particular screen. No privilege required.

d. read-tablet(device-id) - after device has been seized, this call will read a tablet point.

<u>Menu</u> <u>Monitor</u> to

1. Unix

a. seize-device(device-id,privilege) - used to seize control of a device, returns id for usage with system calls. Privilege must be valid for success.

b. temporary-seize-device(device-id,privilege) - used to seize a device temporarily. This is different from the normal seize because Unix remembers who currently has the device and when this module releases the device, Unix restores control to the module who originally seized it.

c. release-device(device-id,privilege) - releases a device. Fails if the privilege is incorrect.

d. point-tablet(device-id) - points the tablet to a particular screen. No privilege required.

e. read-tablet(device-id) - after device has been seized, this call will read a tablet point.

Icon Manager to

1. Stager

a. modify(modification-type,GDS-coord,x-ext,y-ext) the types of modification are blinking and framing. Returns some id for the modification.

b. unmodify(id) - undoes the modification.

c. display-text(string,i-plane-id,x,y,x-ext,y-ext) displays text in the given region. Returns some id.

d. undisplay(id) - undoes the display of text.

e. add-picture(GDS-coord,x-ext,y-ext,picture) - write the picture into the given rectangle.

f. add-text-field(GDS-coord,x-ext,y-ext,text-string) - write text into the given rectangle.

g. erase(GDS-coord,x-ext,y-ext) - erases anything in the given rectangle.

<u>Navigator</u> to

1. joy stick

   a. read-joy-stick() - reads joy stick positions & returns values. Blocks if zero.

2. Stager - note: z is the display's scale factor.

   a. move-rel(dx,dy) - moves screen over i-plane ; returns when complete.

   b. zoom-rel(dz) - performs zooming ; returns when complete.

   c. set-position(I-plane-id,x,y,z)

   d. prepare-for-position(i-plane-id,x,y,z)

   e. reset-modifications() - informs Stager that all modifications are to be undone. Used when a new I-Space is entered.

   f. grant-privilege(device-id,privilege) - passes the privilege to seize a device.

   g. deny-privilege(device-id) - notification the that privilege to seize a device is being removed.

3. Navigational Aids

   a. set-pos(x,y,z) - tells Navigational Aids where user is in the I-Space. From z, the screen size can be determined.

   b. set-up(nav-aid-id,nav-aid-type,aux-screen#)

   c. remove-nav-aid(screen#)

   d. screen-to-GDS-coord(screen-pos,aux-screen#) - used to translate screen coordinates to I-Space coord for a nav aid. Used for rapid transit.

   e. grant-privilege(device-id,privilege) - passes the privilege to seize a device.

   f. deny-privilege(device-id) - notification the that privilege to seize a device is being removed.

4. Unix

   a. seize-device(device-id,prililege) - used to seize control of a device, returns id for usage with system calls. Privilege must be valid for success.

   b. release-device(device-id,privilege) - releases a device. Fails if the privilege is incorrect.

c. read-tablet(device-id) - after device has been seized, this call will read a tablet point.

5. Icon Manager

a. get-icon-pos(icon-id) - returns I-Space position of icon. Used for SQUEL rapid transit.

b. reset-modifications() - clears any modifications. Used when a new I-Space has been entered.

<u>Stager</u> to

1. Display

a. move-rel(dx,dy) -- moves the screen about the display.

b. zoom-rel(dz) - performs the actual zooming.

c. feed-display(rectangles-of-pixels) - feeds new pixels to display.

2. GDS Database

a. get-tile(tile-id)

b. write-tile(tile-id)

<u>Navigational Aids</u>

1. GDS Database & graphics screens

   a. get-i-plane-header(i-plane-id) - gets i-plane header from disk file.

   b. display-image(image-file,screen#) - used to display navigational aid.

   c. draw-rectangle(screen-pos,x-ext,y-ext,color) - draws colored rectangle showing position on a navigational aid.

The following 5 figures show the communication paths between modules. Figure 1 shows the communication paths necessary for controlling the view of the GDS. Figure 2 shows the connections used for processing SQUEL requests. Of course, the connections of Figure 1 would also be used for moving through the GDS, but they are left out since they appear in Figure 1. Figure 3 shows the communication paths for entering and editing ICDL. The connections drawn with dotted lines show that SQUEL may be utilized from the ICDL mode when the user wishes to test an icon class description. Figure 4 contains the connections used by the GDS Editor when it is operating in its full edit mode. Figure 5 shows which modules are called when the Process Overseer changes the mode of operation.

Figure 1

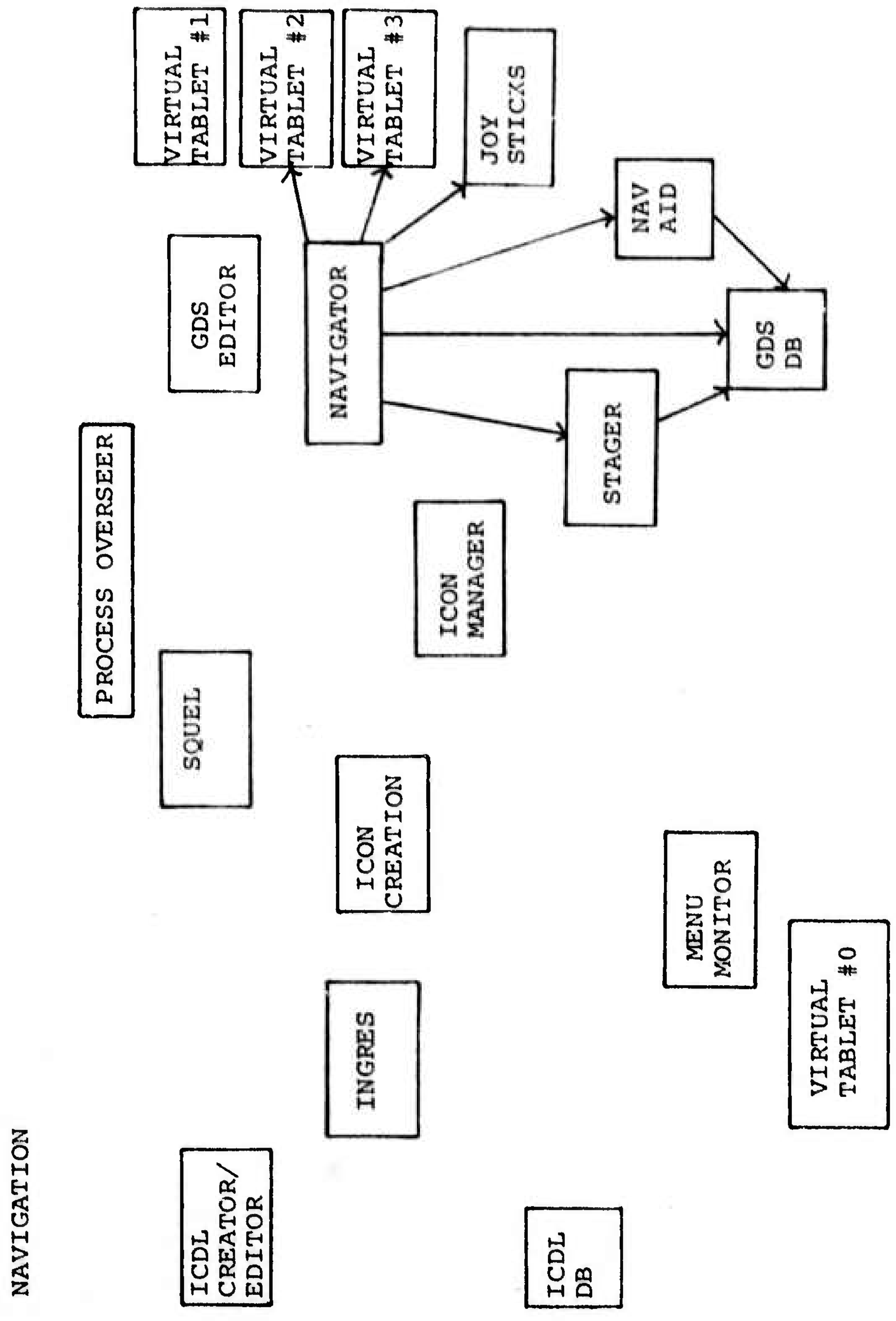

VIRTUAL TABLET #1
VIRTUAL TABLET #2
VIRTUAL TABLET #3
JOY STICKS
NAV AID
GDS EDITOR
NAVIGATOR
GDS DB
STAGER
PROCESS OVERSEER
ICON MANAGER
SQUEL
ICON CREATION
MENU MONITOR
VIRTUAL TABLET #0
INGRES
NAVIGATION
ICDL CREATOR/ EDITOR
ICDL DB

Figure 2

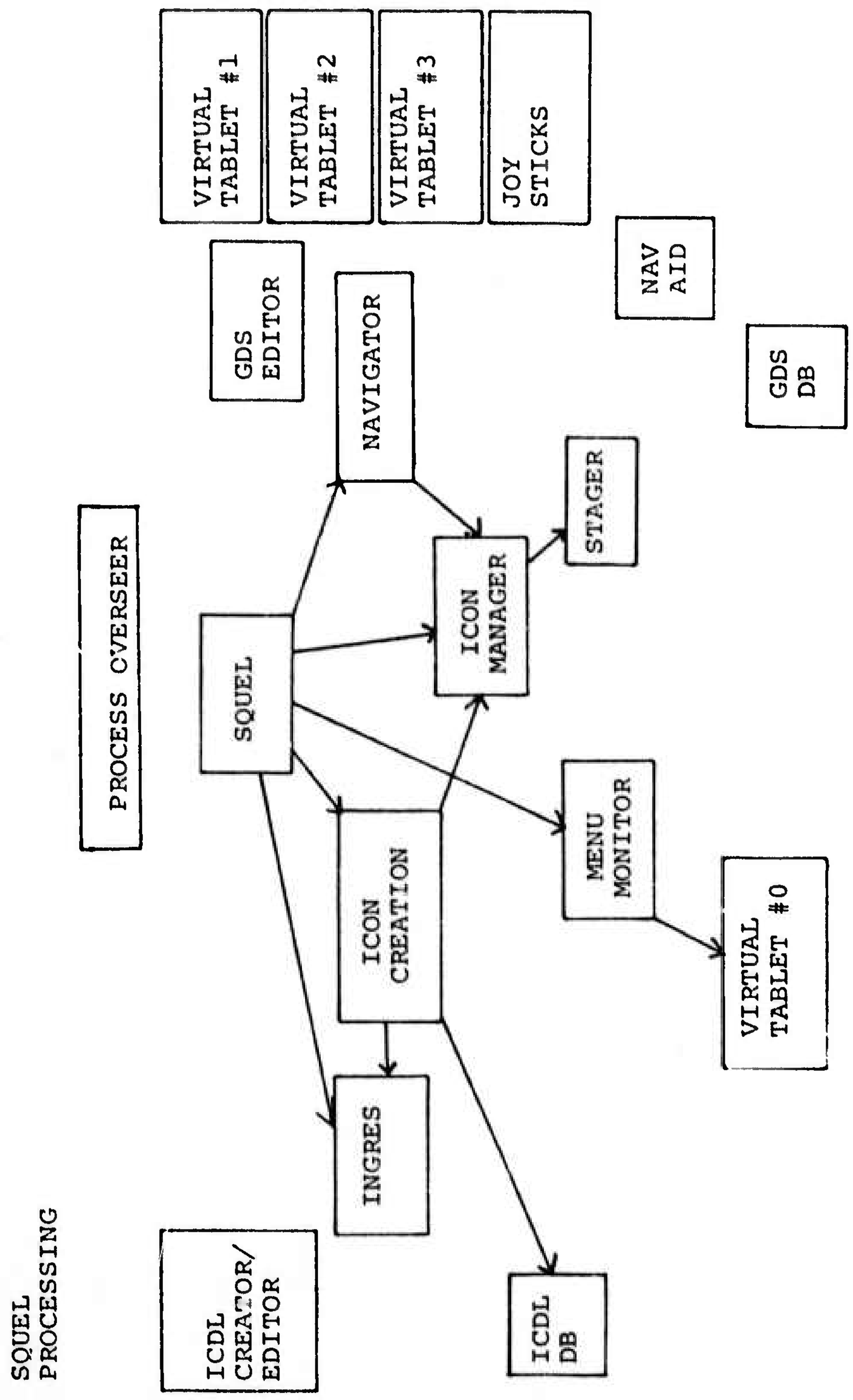

SQUEL PROCESSING
VIRTUAL TABLET #1
VIRTUAL TABLET #2
VIRTUAL TABLET #3
JOY STICKS
NAV AID
GDS EDITOR
NAVIGATOR
GDS DB
STAGER
PROCESS CVERSEER
SQUEL
ICON MANAGER
MENU MONITOR
ICON CREATION
VIRTUAL TABLET #0
INGRES
ICDL CREATOR/ EDITOR
ICDL DB

Figure 3

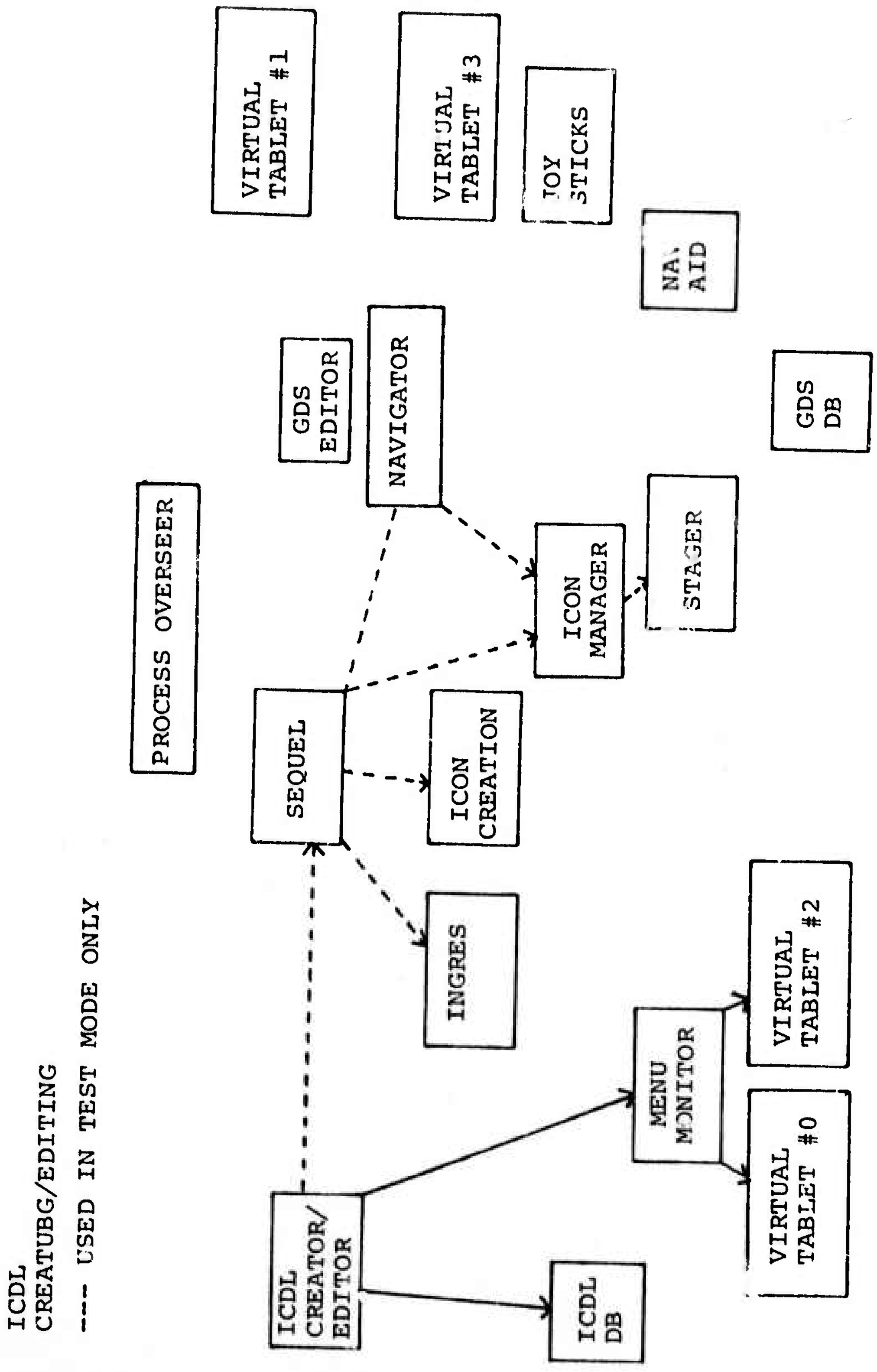

ICDL
CREATUBG/EDITING
--- USED IN TEST MODE ONLY
VIRTUAL TABLET #1
VIRTUAL TABLET #3
JOY STICKS
NA AID
GDS EDITOR
NAVIGATOR
GDS DB
PROCESS OVERSEER
ICON MANAGER
STAGER
SEQUEL
ICON CREATION
INGRES
VIRTUAL TABLET #2
MENU MONITOR
VIRTUAL TABLET #0
ICDL CREATOR/ EDITOR
ICDL DB

Figure 4

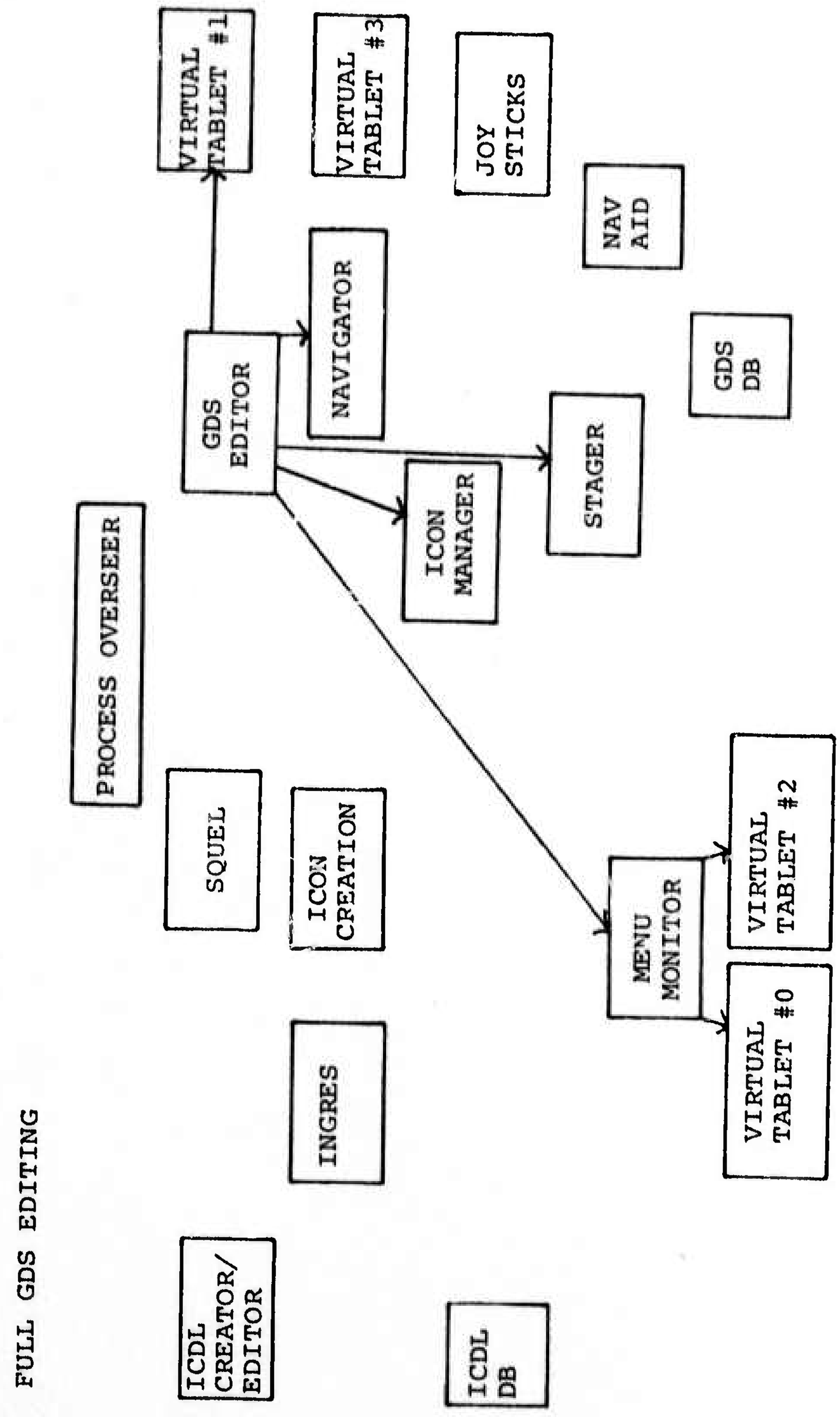
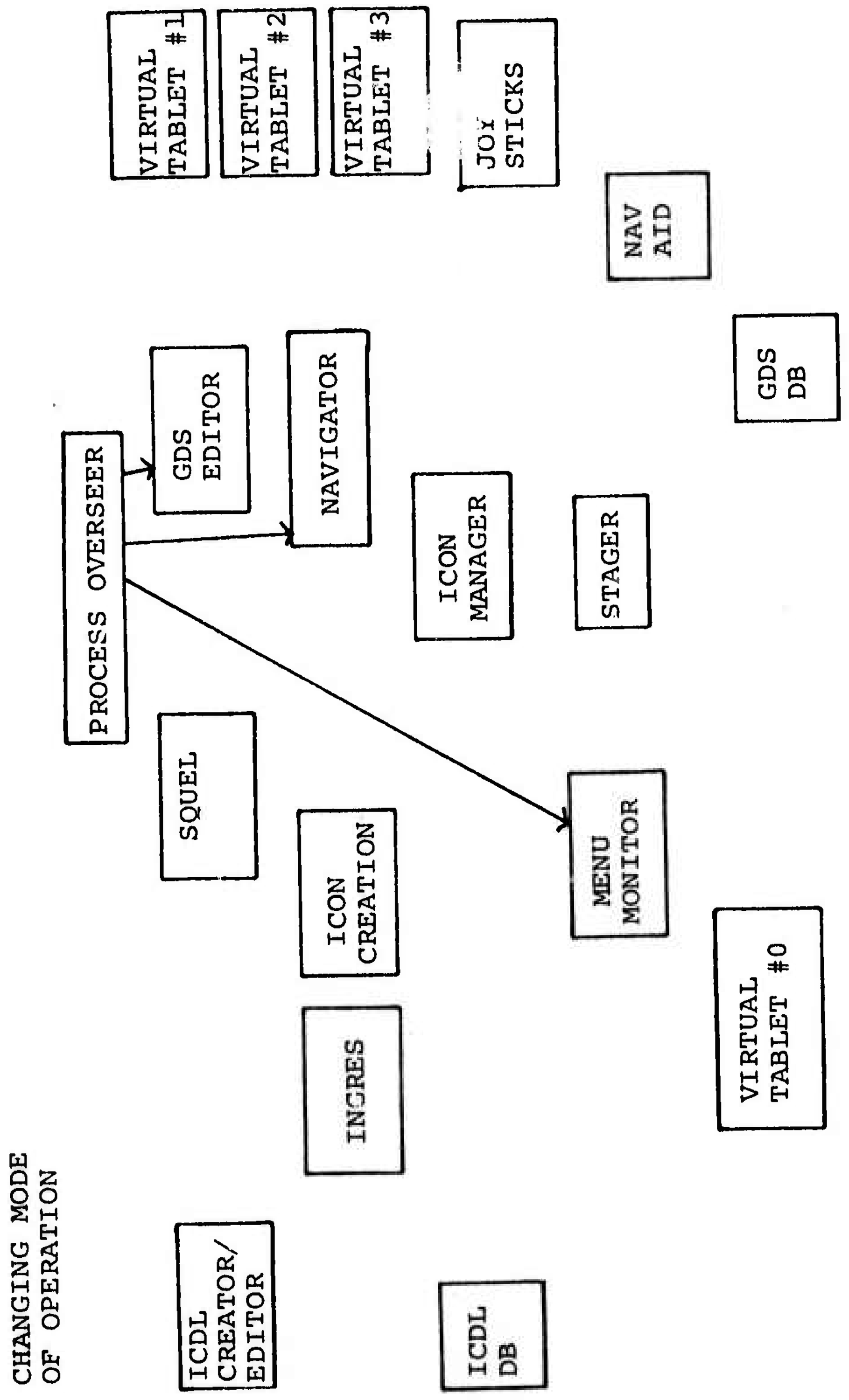

FULL GDS EDITING
VIRTUAL TABLET #1
VIRTUAL TABLET #3
JOY STICKS
NAV AID
NAVIGATOR
GDS EDITOR
GDS DB
STAGER
ICON MANAGER
PROCESS OVERSEER
SQUEL
ICON CREATION
MENU MONITOR
VIRTUAL TABLET #2
VIRTUAL TABLET #0
INGRES
ICDL CREATOR/ EDITOR
ICDL DB

Figure 5


VIRTUAL TABLET #1
VIRTUAL TABLET #2
VIRTUAL TABLET #3
JOY STICKS
NAV AID
GDS DB
GDS EDITOR
NAVIGATOR
ICON MANAGER
STAGER
PROCESS OVERSEER
SQUEL
ICON CREATION
MENU MONITOR
VIRTUAL TABLET #0
INGRES
CHANGING MODE OF OPERATION
ICDL CREATOR/ EDITOR
ICDL DB

References

[HEROT, SCHMOLZE, CARLING, FARRELL]

Herot, Christopher F.; Schmolze, Jim; Carling, Richard; Farrell, Jerry. "Preliminary Design for a Spatial Data Management System", Technical Report CCA-78-09, Computer Corporation of America, 575 Technology Square, Cambridge Massachusetts, June 30, 1978.

[HELD, STONEBRAKER, WONG]

Stonebraker, M.R.; Held, G.D.; Wong, E. "INGRES - A relational data base system", AFIPS Proceedings, Volume 44.